CS540 Exam Review

Deep Learning

Input $X: c_i \times n_h \times n_w$
Kernel $W: c_o \times c_i \times k_h \times k_w \rightarrow$ with bias: $c_o \times (c_i \times k_h \times k_w + 1)$
Output $Y: c_o \times m_h \times m_w$
$m_h = \lfloor (n_h - k_h + p_h + s_h)/s_h \rfloor, m_w = \lfloor (n_w - k_w + p_w + s_w)/s_w \rfloor$

Convolution: multiply kernel over input matrix
kernel value are learned

Pooling: no learnable parameters, apply to each input channel
max: strongest signal in window
avg: average signal strength

Reducing Training Error
Add more layers, increases units in each layer

Reducing Generalization Error
Add more data, augmentation, regularization, weight decay

Residual / Skip

Search

	Complete	Optimal	Time	Space
BFS	Y	?	$O(b^d)$	$O(b^d)$
UCS	Y	Y	$O(C^{*}/\varepsilon)$	$O(b^{C^*/\varepsilon})$
DFS	N	N	$O(b^m)$	$O(bm)$
ID	Y	?	$O(b^d)$	$O(bd)$

Iterative Deepening
search like BFS, fringe is like DFS

A* Search: $g(s) + h(s)$, $h(s) \leq h^*(s)$, $h(s) > 0$
Dominance: $h_1(s) \leq h_2(s) \leq h^*(s)$
h_2 dominates h_1, want h to be as close to h^* as possible

Hill Climbing: find optimal state
move to a neighbor with better $f(s)$
Problems: local optima, defining neighbor
Simulated Annealing: allow downhill moves with
probability $\exp(-\frac{|f(s) - f(t)|}{T})$, T decreases over time

Game

Nash equilibrium

- Strictly dominant: a_i strictly better than a_i'
- a^* is a Nash equilibrium if no player has incentive to unilaterally deviate
- Mixed strategies: can randomize actions
- Every finite game has at least 1 Nash equilibrium but not necessarily pure

Minimax

- Max: take max of children
- Min: take min of children
- Pruning: get rid of bad branches
- Heuristics: long game might require large tree
  - limit search depth, use heuristic function

Reinforcement Learning

Markov Decision Process

- Markov assumption: transition probability only depends on s_t and a_t and not earlier history
- Bellman Equation: $V^*(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s,a) V^*(s')$

Q-learning: get an action-utility function $Q^*(s,a)$

- $\pi^*(s) = \arg\max_a Q^*(s,a)$
- Epsilon-greedy: with $0<\epsilon<1$ probability, take a random action
  - Allows balance between exploration & exploitation
- Update function:
  $Q(s_t,a_t) \leftarrow (1-\alpha)Q(s_t,a_t) + \alpha(r_t + \gamma \max_b Q(s_{t+1},b))$

Probability

Bayesian Inference: $\dfrac{\overbrace{P(E|H)}^{\text{likelihood}}\overbrace{P(H)}^{\text{Prior}}}{P(E)}$

Law of Total Probability: $P(A) = \sum_{b \in B} P(A, b) = \sum_{b \in B} P(A|b)P(b)$

Naive Bayes: $P(e_1, e_2, \ldots e_d | H) = P(e_1|H)P(e_2|H)\ldots P(e_d|H)$

Conditional Independence Assumption

feature attributes conditionally independent on label

CS 540 Midterm Cheat Sheet

Probability
- language to express uncertainty

Given random variables X, Y:

joint distribution: $P(X=a, Y=b)$

marginal distribution: $P(X=a)$

independence: $P(X,Y)=P(X)P(Y)$

conditional probability: $P(X=a|Y=b)$
$= P(X=a, Y=b)/P(Y=b)$

Conditional independence: $P(X,Y|Z)$
$= P(X|Z)\,P(Y|Z)$

chain rule: $P(A_1, A_2, \dots, A_n)$
$= P(A_1)P(A_2|A_1)\dots P(A_n|A_{n-1},\dots,A_1)$

Bayes Rule: $P(H|E) = \frac{P(H,E)}{P(E)} = \frac{P(E|H)P(H)}{P(E)}$

$H \rightarrow$ hypothesis
$E \rightarrow$ evidence.

prior: $P(H)$ estimate of probability w/o evidence

likelihood: $P(E|H)$ probability of evidence given a hypothesis

posterior: $P(H|E)$ probability of hypothesis given hypothesis

posterior = likelihood × prior

Natural Language Processing
- Use probalistic models to assign a probability to a sentence

$P(w_1, w_2, \dots w_n) = P(w_1)P(w_2|w_1)$
$\dots P(w_n|w_{n-1}\dots w_1)$

Markov-Type Assumptions

$P(w_i|w_{i-1}w_{i-2}\dots w_1) = P(w_i|w_{i-1}w_{i-2}\dots w_{i-k})$

$k=0$: unigram → full independence

$k=1$ bigram → Markov chain

$k=n-1$: n-gram

Issues: 1. Multiply tiny numbers
use log, add instead of multiply
2. n-grams with 0 probability
use smoothing, eg: add-1

Evaluate: 1. Extrinsic Evaluation
compare 2 models
2. Perplexity
$PP(W) = P(w_1, w_2, \dots w_n)^{-\frac{1}{n}}$

Representing words
- one-hot vectors
- word embeddings

Linear Algebra & PCA
- building blocks for all models
- vectors, matrices
- eigenvalues, eigenvectors:
soln's to $Av = \lambda v$

Principal Component Analysis (PCA)
reduce dimensions
- when data is approx lower dim
- find axes of subspace
orthogonal directions
$v_1, \dots v_r \in \mathbb{R}^d$
- v's are eigenvectors of X^TX.
- X^TX proportional to sample covariance matrix
- PCA is eigendecomposition of sample covariance matrix

Machine Learning
1. Supervised Learning
   - Classification: label is discrete
   - Regression: label is continuous

   learn a function: $f: X \rightarrow Y$
   to predict label y on future data x
2. Unsupervised Learning
   discover interesting patterns and structure in data
   - Clustering (k-means, hierarchical, Bayesian, graph-based)
   - Visualization (t-SNE)
   - Density Estimation (kernel density estimation)
3. Reinforcement Learning
   Given agent that can take actions & reward function
   Goal: learn to choose actions that maximize future reward total

Unsupervised Learning: Clustering

- K-means
  1. Randomly pick k cluster centers
  2. Find closest center for each point
  3. Update cluster by computing centroids

  Repeat until convergence
- Hierarchical
  - Agglomerative: bottom up
    - merge pairs of clusters closest until all clusters merged
    - single linkage: closest points in clusters
    - complete linkage: furthest points in clusters
  - Divisive: top down
    - all points in one cluster, split clusters
- Spectral
  1. Compute Laplacian
  2. Compute k smallest eigenvectors
  3. Set U as n×k matrix with $u_1, \dots, u_k$ as cols
  4. Run k-means

  handles intuitive separation

Linear Models & Linear Regression

Predict linear combination of x components + intercept: $f(x) = \theta_0 + \theta_1 x_1 + \dots + \theta_d x_d$

loss function: mean square error

$\frac{1}{n}||X\theta - y||^2$

- optimize with SGD:
  - calculate partial derivatives
  - update θ until values converge

Closed form solution:

$\hat{\theta} = (X^T X)^{-1} X^T y$

Logistic Regression → classification

convert $\theta^T x$ to probability in [0, 1]

$p(y=1|x) = \frac{1}{1+\exp(-\theta^T x)}$

If θ^T is big → $\exp(-\theta^T x)$ small → p close to 1

If $\exp(-\theta^T x)$ big → p close to 0

KNN & Naive Bayes

K-Nearest Neighbors

- compute decision boundary based on k-nearest neighbor labels
- As k↑, flexibility↓, sensitivity↓

Maximum Likelihood Estimation

best fits the data

$\hat{y} = \hat{f}(x) = \arg\max p(y|x)$

$= \arg\max_y p(x|y)\,p(y)$

$\hat{y}$ → prediction

$p(y|x)$ → posterior

Naive Bayes

assumption: conditional independence of feature attributes

Neural Networks

Perceptron: $o = \sigma(\langle w, x\rangle + b)$

XOR Problem: Single perceptron can only draw linear decision boundary

σ → activation function

sigmoid, ReLU, tanh

Multi-layer perceptron: solves XOR

Softmax: turn output into probability

Training: calculate a loss

softmax loss, cross-entropy loss

Minibatch SGD: calculate gradient descent on subset

back propagation: update weights

gradient vanishing

gradient exploding

training error: overfitting, underfitting

model capacity: ability to fit a variety of models

regularization: techniques for better generalization

weight decay

drop out